Scalable Computing Challenges: An Overview

Michael A. Heroux
Sandia National Laboratories

•**Special Thanks:**
- **DOE IAA**
- **LDRD**
- **NNSA ASC**

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Four Challenges

- Parallel Programming Transformation
  - MPI+Serial ➔ …
  - Goal: 1-10 Billion-way parallel.
- Beyond the Forward Problem
  - Optimal, bounded solutions
  - New linear algebra kernels.
- Fault-resilient application execution
  - Progress in the presence of system instability
- High quality, multi-institutional, multi-component, multi-layered SW environment.
  - Single monolithic application ➔ …

Sandia National Laboratories

Preliminaries

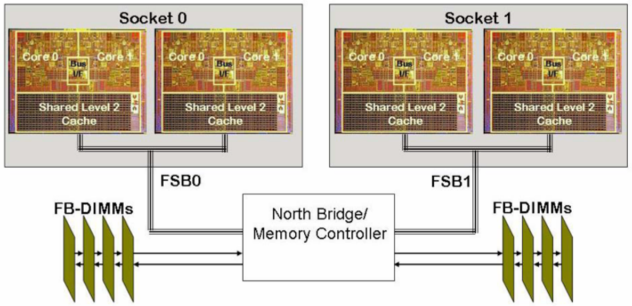
About MPI

- MPI will be the primary inter-node programming model.
- Right ingredients:
  - Portable, ubiquitous.
  - Forced alignment of work/data ownership and transfer.
- Matches architectures:
  - Interconnects of best commercial node parts.
- **Key point: Very few people write MPI calls.**
  - Domain-specific abstractions.
  - Example: Epetra_MpiDistributor
    - 20 revisions since initial checkin in December 2001.
    - Only three developers made non-trivial changes in 8+ years.
    - No nontrivial changes in 4+ years. No changes in 2+ years.
- New languages:
  - Big fan of Co-Array Fortran (Have been for 15 years: F--).
  - Chapel looks good.
  - But tough uphill climb.
- Real question: How do we program the node?

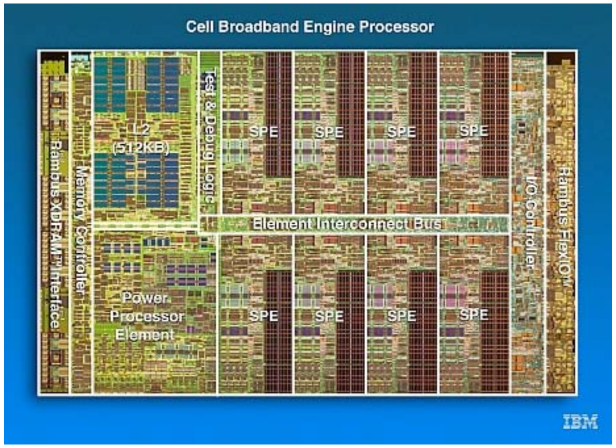
Sandia National Laboratories

Node Classification

- Homogeneous multicore:
  - SMP on a chip.
  - NUMA nodes.
  - Varying memory architectures.



- Heterogeneous multicore:
  - Serial/Controller processor(s).
  - Team of identical, simpler compute processors.
  - Varying memory architectures.



Why Homogeneous vs. Heterogeneous?

- Homogeneous:
  - Out-of-the-box: Can attempt single-level MPI-only.
  - m nodes, n cores per node: p = m*n
  - mpirun -np p …
- Heterogeneous:
  - Must think of compute cores as "co-processors".
  - mpirun -np m …
  - Something else on the node.
- Future:
  - Boundary may get fuzzy.
  - Heterogenous techniques can work well on homogeneous nodes.

Sandia National Laboratories

Single Core Performance: Still improving for some codes

- MiniFE microapp.
- Clock speeds stable: ~ 2GHz.
- FP-friendly computations stalled.
- Memory-intensive computations still improving.
- Prediction: Memory bandwidth “wall” will fall.

Year	Processor	Clock (GHz)	Cores/ socket	MFLOPS/ sec
2003	AMD Athlon	1.9	1	178
2004	AMD Opteron	1.6	1	282
2005	Intel Pentium M	2.1	1	310
2006	AMD Opteron	2.2	2	359
2007	Intel Woodcrest	1.9	4	401
2007	AMD Opteron	2.1	4	476
2007	Intel Core Duo	2.3	2	508
2008	AMD Barcelona	2.1	4	550
2009	Intel Nehalem	2.2	4	~900

Sandia National Laboratories

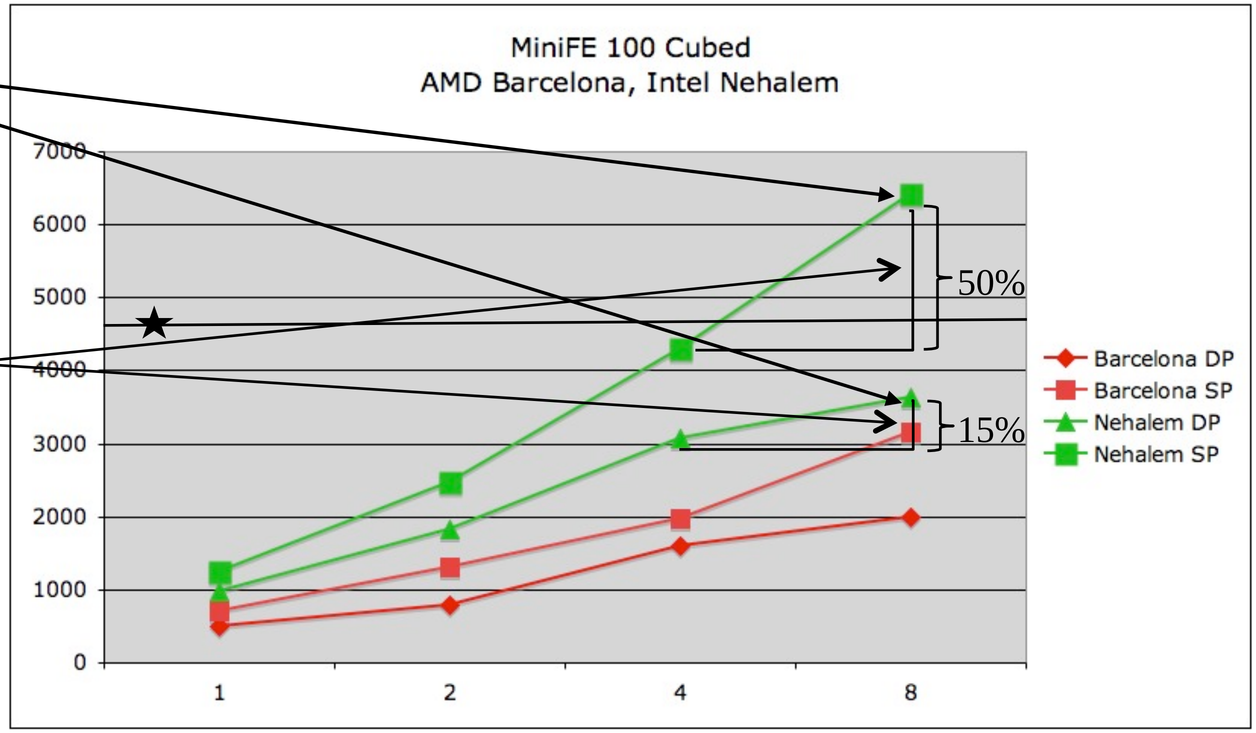
Mixed Precision: Now is the Time

with Chris Baker, Alan Williams, Carter Edwards

Sandia National Laboratories

The Case for Mixed Precision

- Float useful:
  - Always true.
  - More important now.
  - Mixed precision algorithms.
- Bandwidth even more important:
  - Saturation means loss of effective core use.
  - Loss of scaling opportunity for modern systems.



- Mixed precision & GPUs:
  - GEForce GTX280
    - SP: 624 GFLOPS/s
    - DP: 78 GFLOPS/s
- First MiniFE result on GPUs: 4.71 GFLOP/s (SP)
- Expected results: 12 GFLOP/s (SP), 6 GFLOP/s (DP)

Sandia National Laboratories

C++ Templates

How to implement mixed precision algorithms?

- C++ templates only sane way.
- Moving to completely templated Trilinos libraries.
- Core Tpetra library working.
- Other important benefits.

Template Benefits:

– Compile time polymorphism.
– True generic programming.
– No runtime performance hit.
– Strong typing for mixed precision.
– Support for extended precision.
– Many more…

Template Drawbacks:

– Huge compile-time performance hit:
  - But this is OK: Good use of multicore :)
  - Can be greatly reduced for common data types.
– Complex notation (for Fortran & C programmers).

C++ Templates and Multi-precision

```
// Standard method prototype for apply matrix-vector multiply:
template<typename ST, typename OT>
CrsMatrix::apply(Vector<ST, OT> const& x, Vector<ST, OT>& y)
```

```
// Mixed precision method prototype (DP vectors, SP matrix):
template<typename ST, typename OT>
CrsMatrix::apply(Vector<ScalarTraits<ST>::dp(), OT>  const& x,
                 Vector<ScalarTraits<ST>::dp(), OT> & y)
```

```
// Sample usage:
Tpetra::Vector<double, int> x, y;
Tpetra::CrsMatrix<float, int> A;
A.apply(x, y);  // Single precision matrix applied to double precision vectors
```

Tpetra Linear Algebra Library

- Tpetra is a templated version of the Petra distributed linear algebra model in Trilinos.
  - Objects are templated on the underlying data types:

```
MultiVector<scalar=double, local_ordinal=int,
            global_ordinal=local_ordinal> …
CrsMatrix<scalar=double, local_ordinal=int,
            global_ordinal=local_ordinal> …
```

  - Examples:

```
MultiVector<double, int, long int> V;
CrsMatrix<float> A;
```

Speedup of float over double in Belos linear solver.

float	double	speedup
18 s	26 s	1.42x

Scalar	float	double	double-double	quad-double
Solve time (s)	2.6	5.3	29.9	76.5
Accuracy	10^{-6}	10^{-12}	10^{-24}	10^{-48}

Arbitrary precision solves using Tpetra and Belos linear solver package

Sandia National Laboratories

FP Accuracy Analysis:
FloatShadowDouble Datatype

```
class FloatShadowDouble {

public:
  FloatShadowDouble( ) {
    f = 0.0f;
    d = 0.0;  }
  FloatShadowDouble( const FloatShadowDouble & fd) {
    f = fd.f;
    d = fd.d;  }
…
inline FloatShadowDouble operator+= (const FloatShadowDouble & fd ) {
    f += fd.f;
    d += fd.d;
    return *this;  }
…
inline std::ostream& operator<<(std::ostream& os, const FloatShadowDouble& fd) {
  os << fd.f << "f " << fd.d << "d";  return os;}
```

- Templates enable new analysis capabilities
- Example: Float with “shadow” double.

FloatShadowDouble

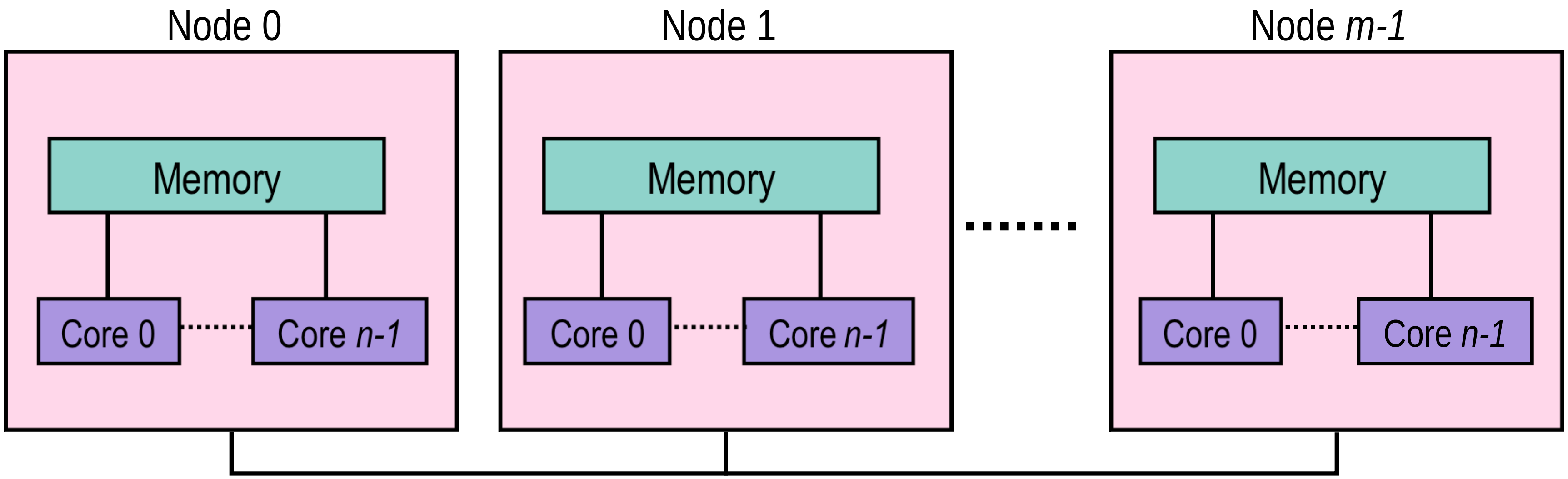
```
Sample usage:
#include "FloatShadowDouble.hpp"
Tpetra::Vector<FloatShadowDouble> x, y;
Tpetra::CrsMatrix<FloatShadowDouble> A;
A.apply(x, y);  // Single precision, but double results also computed, available
```

Initial Residual =		455.194f	455.194d
Iteration = 15	Residual =	5.07328f	5.07618d
Iteration = 30	Residual =	0.00147f	0.00138d
Iteration = 45	Residual =	5.14891e-06f	2.09624e-06d
Iteration = 60	Residual =	4.03386e-09f	7.91927e-10d

Programming Models for Scalable Homogeneous Multicore (beyond single-level MPI-only)

Sandia National Laboratories

Parallel Machine Block Diagram

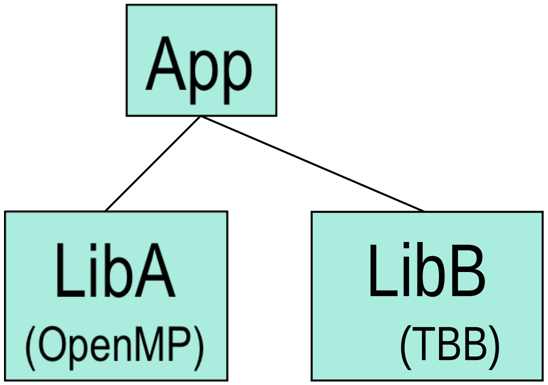


- Parallel machine with $p = m * n$ processors:
  - m = number of nodes.
  - n = number of shared memory processors per node.
- Two ways to program:
  - **Way 1**: p MPI processes.
  - **Way 2**: m MPI processes with n threads per MPI process.
- New third way:
  - "Way 1" in some parts of the execution (the app).
  - "Way 2" in others (the solver).

Sandia National Laboratories

Threading under MPI

- Default approach: Successful in many applications.
- Concerns:
  - Opaqueness of work/data pair assignment.
    - Lack of granularity control.
  - Collisions: Multiple thread models.
    - Performance issue, not correctness.



- Bright spot: Intel Thread Building Blocks (TBB).
  - Iterator (C++ language feature) model.
  - Opaque or transparent: User choice.

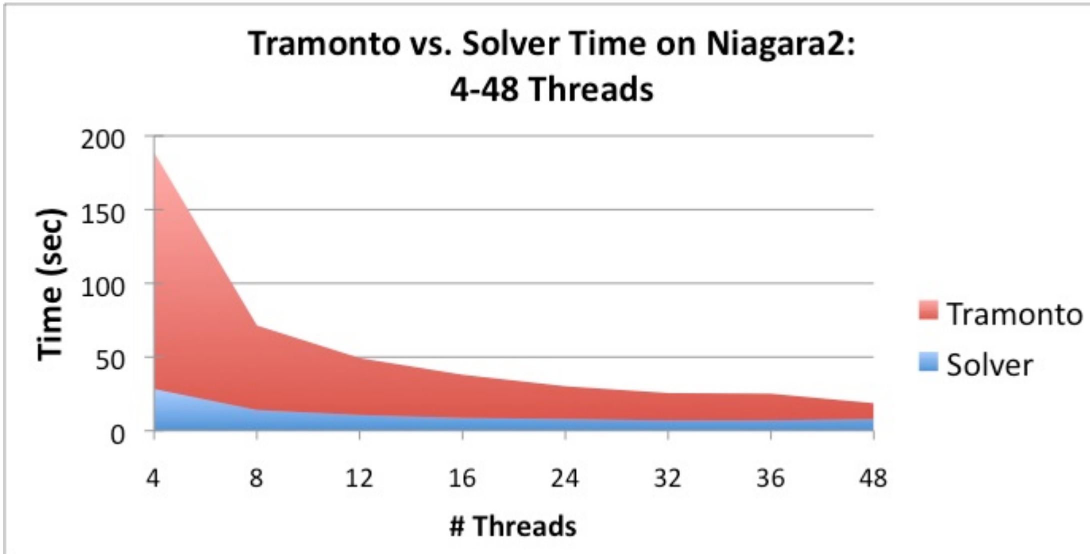
MPI Under MPI

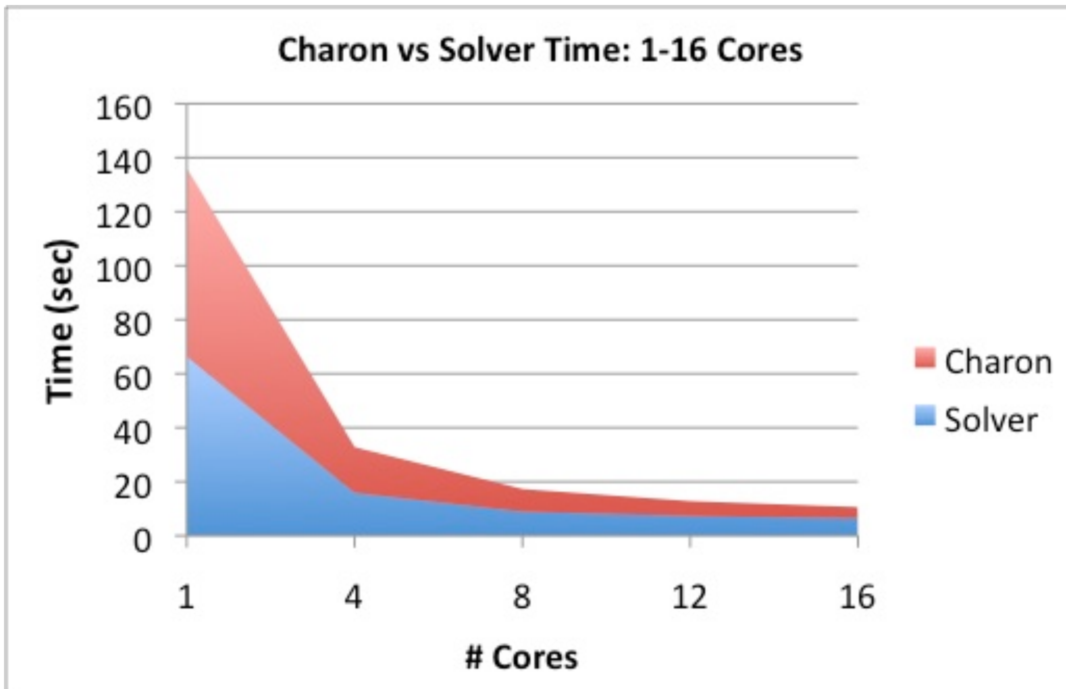
- Scalable multicores:
  - Two different MPI architectures.
  - Machines within a machine.
- Exploited in single-level MPI:
  - Short-circuited messages.
  - Reduce network B/W.
  - Missing some potential.
- Nested algorithms.
- Already possible.
- Real attraction: No new node programming model.
- Can even implement shared memory algorithms (with some enhancements to MPI).

"Ping-pong" test	Latency (microsec)	Bandwidth (MB/sec)
Inter-node machine	0.71	1082
Intra-node machine	47.5	114

Sandia National Laboratories

Multicore Scaling: App vs. Solver





Application:

- Scales well (sometimes superlinear)
- MPI-only sufficient.

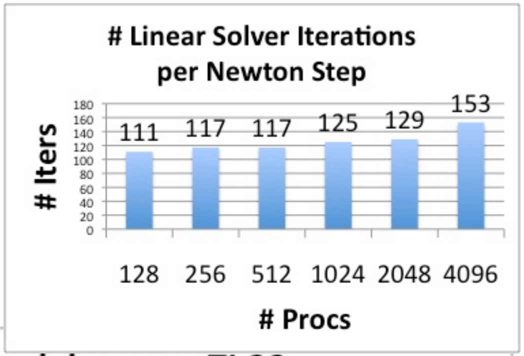
Solver:

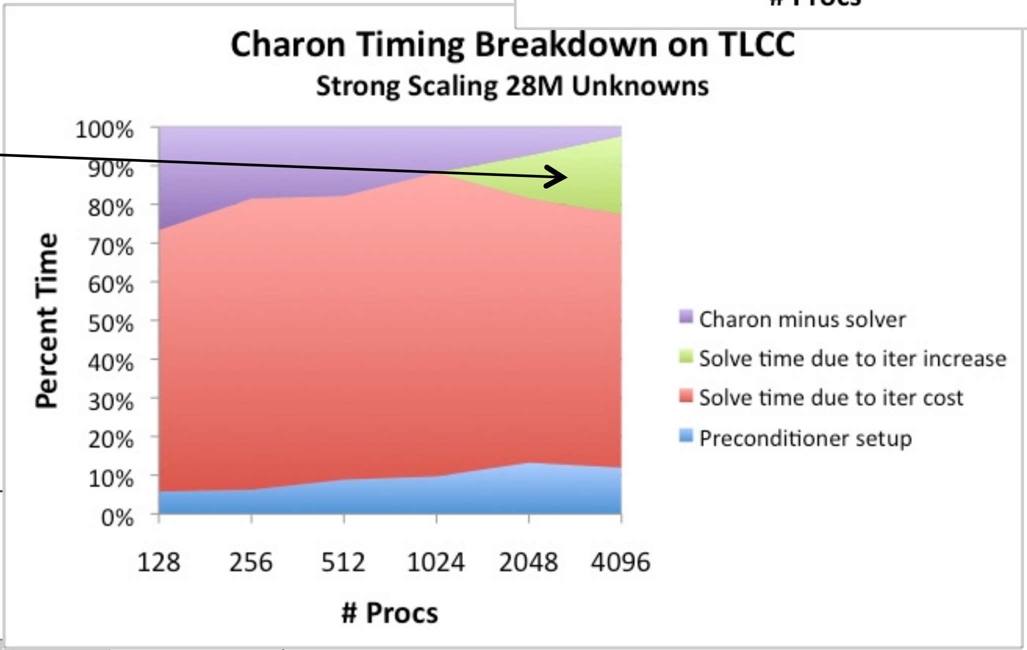
- Scales more poorly.
- Memory system-limited.
- MPI+threads can help.

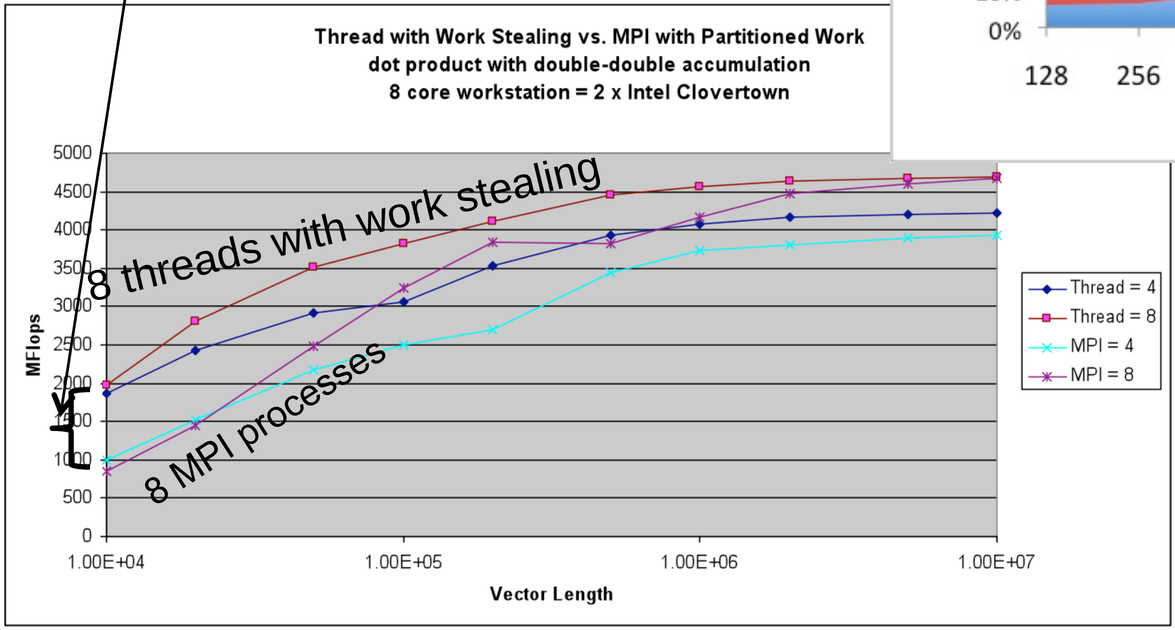
*** Charon Results: Lin & Shadid TLCC Report**

Hybrid Parallelism Opportunities

- **Selective Shared Memory Use**:
  - App: 4096 MPI tasks.
  - Solver: 256 MPI tasks, 16-way threading.
- **Robustness:**
  - 117 iterations (not 153).
  - Eliminates green region.
- **Speed: Threading (carefully used):**
  - Same asymptotic speed as MPI.
  - Faster ramp up: 2X or more.
  - Better load imbalance tolerance.







Bottom line: Hybrid parallelism promises better:

- Robustness,
- Strong scaling and
- Load balancing.

* Thread Results:
H. Carter Edwards

Sandia National Laboratories

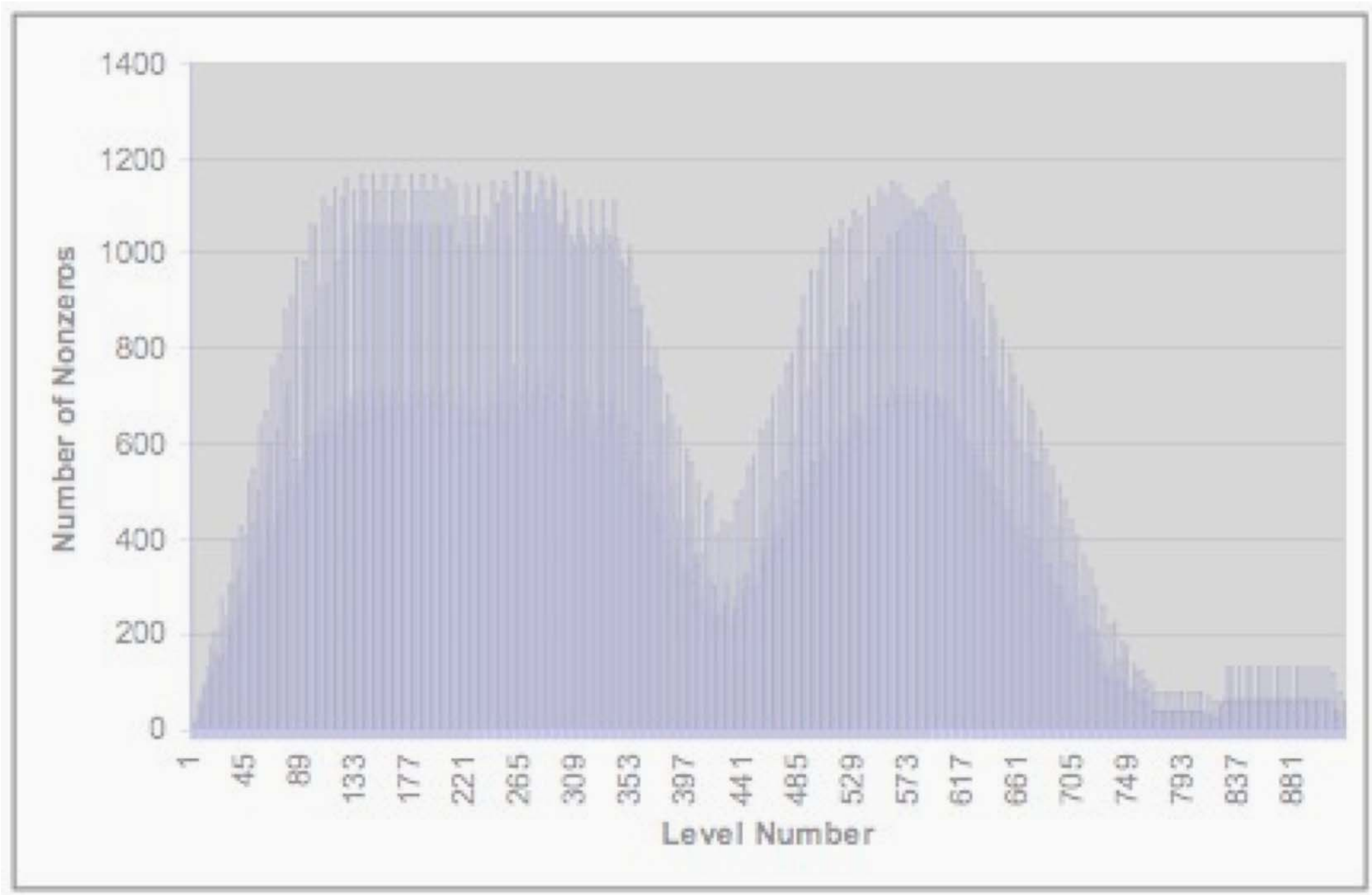
$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & 0 & 1 & 0 & 0 \\ 0 & 0 & l_{43} & 1 & 0 \\ l_{51} & 0 & l_{53} & 0 & 1 \end{bmatrix}$$

Hybrid Parallelism: Shared Memory Algorithms

Solve $Ly = x$.

Critical kernel for many scalable preconditioners.

Key Idea: Use sparsity as resource for parallelism.



Sandia National Laboratories

Heterogeneous Multicore Issues

Sandia National Laboratories

Excited about multimedia processors

- Inclusion of native double precision.
- Large consumer market.
- Qualitative performance improvement over standard microprocessors…
- If your computation matches the architecture.
- Many of our computations do match well.
- Homogeneous vs. Heterogeneous: Indistinguishable in Future.

APIs for Heterogeneous Nodes
(A mess, but some light)

Processor	API
NVIDIA	CUDA
AMD/ATI	Brook+
STI Cell	ALF
Intel Larrabee	Ct
Most/All?	Sequoia
Most	RapidMind (Proprietary)
Apple/All	OpenCL

Commonality: Fine-grain functional programming.

Our Response: A Library Node Abstraction Layer

Sandia National Laboratories

Preparing for Manycore

Refactoring for Manycore

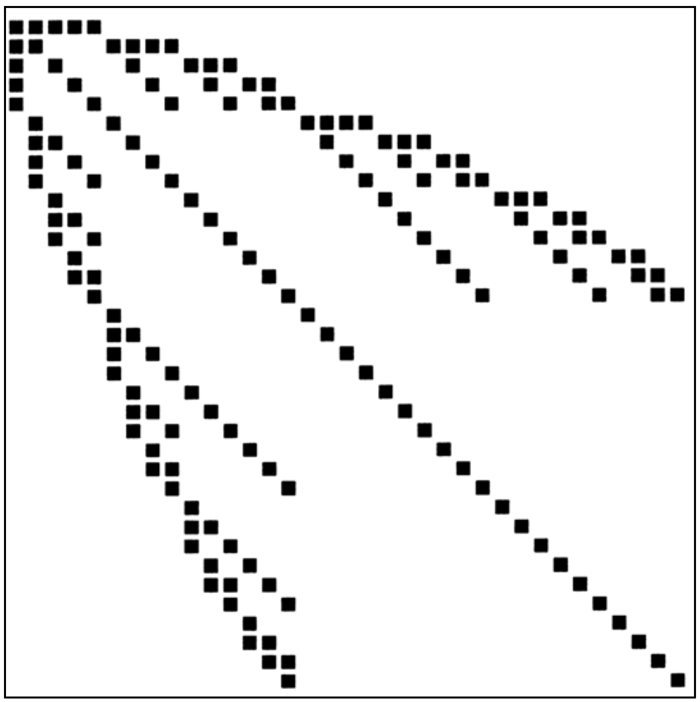
- Regardless of node-level programming model:
  - Isolate all computation to stateless functions.
  - Formulate functions so that work granularity can vary.
- Fortran/C:
  - Natural approach.
  - Still requires some change for variable granularity.
- C++:
  - Separate data organization from functions.
  - Can still have computational methods.

Beyond the Forward Problem

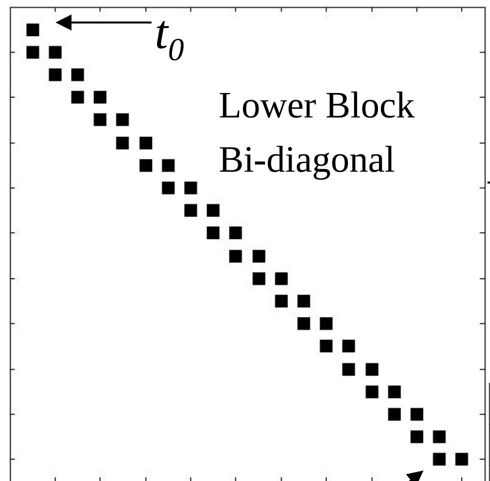
Advanced Modeling and Simulation Capabilities: Stability, Uncertainty and Optimization

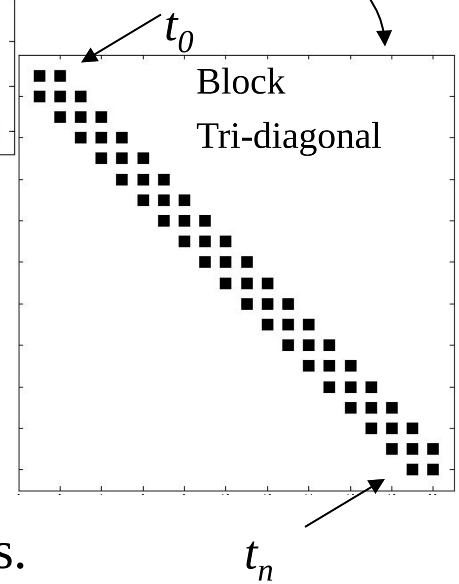
- Promise: 10-1000 times increase in parallelism (or more).

SPDEs:



Transient Optimization:





- Pre-requisite: High-fidelity "forward" solve:
  - Computing families of solutions to similar problems.
  - Differences in results must be meaningful.

■ - Size of a single forward problem

Advanced Capabilities: Readiness and Importance

Modeling Area	Sufficient Fidelity?	Other concerns	Advanced capabilities priority
Seismic *S. Collis, C. Ober*	Yes.	None as big.	Top.
Shock & Multiphysics (Alegra) *A. Robinson, C. Ober*	Yes, but some concerns.	Constitutive models, material responses maturity.	Secondary now. Non-intrusive most attractive.
Multiphysics (Charon) *J. Shadid*	Reacting flow w/ simple transport, device w/ drift diffusion, …	Higher fidelity, more accurate multiphysics.	Emerging, not top.
Solid mechanics *K. Pierson*	Yes, but…	Better contact. Better timestepping. Failure modeling.	Not high for now.

Sandia National Laboratories

Advanced Capabilities: Other issues

- Non-intrusive algorithms (e.g., Dakota):
  - Task level parallel:
    - A true peta/exa scale problem?
    - Needs a cluster of 1000 tera/peta scale nodes.
- Embedded/intrusive algorithms (e.g., Trilinos):
  - Cost of code refactoring:
    - Non-linear application becomes “subroutine”.
    - Disruptive, pervasive design changes.
- Forward problem fidelity:
  - Not uniformly available.
  - Smoothness issues.
  - Material responses.

Sandia National Laboratories

Advanced Capabilities: Derived Requirements

- Large-scale problem presents collections of related subproblems with forward problem sizes.

- Linear Solvers: $Ax = b \rightarrow AX = B, \;\; Ax^i = b^i, \;\; A^i x^i = b^i$
  - Krylov methods for multiple RHS, related systems.

- Preconditioners: $A^i = A_0 + \Delta A^i$
  - Preconditioners for related systems.

- Data structures/communication: $pattern(A^i) = pattern(A^j)$
  - Substantial graph data reuse.

Fault Resilience

with Patty Hough, Vicki Howle

Sandia National Laboratories

Soft errors are becoming more prevalent due to small features operating at low voltages

- “At 8 nm process technology, it will be harder to tell a 1 from a 0.” (Camp, 2008)
- …
- Soft errors are scary to apps
  - Computation proceeds but is wrong
  - Careful verification required
  - What if verification has soft errors?

Sandia National Laboratories

Users' View of the System Now vs. Future

- Now:
  - "All nodes up and running."
  - Certainly nodes fail, but invisible to user.
- Future:
  - Nodes in one of four states.
    - Dead.
    - Dying (perhaps producing faulty results).
    - Reviving.
    - Running properly (hopefully large portion).
  - Not hidden from user.

Sandia National Laboratories

Consider GMRES as an example of how soft errors affect correctness

- Basic Steps
  1) Compute Krylov subspace (sparse matrix-vector multiplies)
  2) Compute orthonormal basis for Krylov subspace (matrix factorization)
  3) Compute vector yielding minimum residual in subspace (linear least squares)
  4) Map to next iterate in the full space
  5) Repeat until residual is sufficiently small

- More examples in Bronevetsky & Supinski, 2008

Sandia National Laboratories

Every calculation matters

- Small PDE Problem: Dim 21K, Nz 923K.
- ILUT/GMRES
- Correct computation 35 Iters: 343M FLOPS
- Two examples of a single bad floating point op

Description	Iterations	FLOPS	Recursive Residual Error	Solution Error
All Correct Calcs	35	343M	4.6e-15	1.0e-6
Iter=2, y[1] += 1.0 SpMV incorrect Ortho subspace	35	343M	6.7e-15	3.7e+3
Q[1][1] += 1.0 Non-ortho subspace	N/C	N/A	7.7e-02	5.9e+5

Sandia National Laboratories

One possible approach is transactional computation

- Database transactions: atomic
- Transactional memory: atomic memory operation
- Transactional computation:
  - Designated sensitive computation region (orthogonalization step in GMRES)
  - Guarantee accurate computation or notify user

Sandia National Laboratories

Needs to be coupled with guaranteed data regions

- User-designated reliable data region
- Extra protection to improve reliable data storage and transfer
- Examples
  - Original input data (needed for verification)
  - Linear solver: *A*, *x*, *b*
  - Orthogonal vectors for GMRES

Sandia National Laboratories

More generally, what should application developers do?

- Abandon the assumption that the system can continue to guarantee reliability and correctness???
- Work with system, system software, middleware, etc. developers to learn what can be provided and to develop requirements
- Develop a more holistic view of application development – develop algorithms/applications suitable for running correctly through failure and handling multi-threading
- Reserve the right to use slower, more reliable systems

Software Issues

Sandia
National
Laboratories

Barely Sufficient Software Engineering:
Ten SW Engineering Practices

0 Manage source (the basics)

1 Use issue-tracking software for requirements, features and bugs

2 Manage source (beyond the basics)

3 Use mail lists to communicate

4 Use checklists for repeated processes

5 Create barely sufficient, source-centric documentation

6 Use build-configuration management tools

7 Write tests first, run them often

8 Program tough stuff together

9 Use a formal release process

10 Perform continual process improvement

About “Barely sufficient”

- **A minimalist attitude to formal processes:**
  - **Adopt only those that have a large impact.**
- **Mindless Imposition of Formal SE bad for CSE community:**
  - **Large-scale formal document generation as “first step”.**
  - **Large effort to satisfy an external requirement, does not benefit the project team.**
  - **Documents become out-of-date quickly and therefore are irrelevant or even misleading.**

- **Formal documents:**
  - **Certainly play a role in a project:**
    - **Domain vision statement, e.g., Trilinos Strategic Goals.**
    - **Highlighted core, ACM TOMS article *An Overview of the Trilinos Project*.**
  - **Modest, should be developed after the product architecture is stable.**
  - **Are essential when a product is ready for hand-off to maintenance team.**

Sandia National Laboratories

Summary
Four Challenges → Opportunities

- Parallel Programming Transformation
  - Start now: Refactor using functional programming.
  - Develop your own Node API (or consider ours).
- Beyond the Forward Problem
  - Plenty of parallelism. Lots of work.
  - New collection of linear problems to solve.
- Fault-resilient application execution
  - New opportunities to reformulate core algorithms.
- High quality, multi-institutional, multi-component, multi-layered SW environment.
  - Time to start (continue) SW engineering efforts.

Sandia National Laboratories